MIE 1612: Stochastic Programming and Robust Optimization

Fall 2019

Syllabus

Instructor: Prof. Merve Bodur
Office: BA8106
Office hour: Wednesday 2-3pm (or by appointment)
E-mail: bodur@mie.utoronto.ca
P.S. Please include course code (MIE 1612) in your email subject

TA: Maryam Daryalal
Office hour: Monday 2:30-3:30pm, in BA8119 (or by appointment)
E-mail: m.daryalal@mail.utoronto.ca
P.S. Please include course code (MIE 1612) in your email subject

Lectures: Monday 4-6pm (GB119) and Wednesday 1-2pm (GB303)

Suggested texts:

Lectures on Stochastic Programming – Modeling and Theory, SIAM, Shapiro, Dentcheva, and Ruszczyński, 2009.

Introduction to Stochastic Programming, Springer-Verlag, Birge and Louveaux, 2011.

Stochastic Programming, Wiley, Kall and Wallace, 1994.

Stochastic Programming, Handbooks in OR and MS, Elsevier, Ruszczyński and Shapiro, 2003.

Robust Optimization, Princeton University Press, Ben-Tal, El Ghaoui, and Nemirovski, 2009.

Course web pages: Quercus, Piazza

Official course description: Stochastic programming and robust optimization are optimization tools dealing with a class of models and algorithms in which data is affected by uncertainty, i.e., some of the input data are not perfectly known at the time the decisions are made. Topics include modeling uncertainty in optimization problems, two-stage and multistage stochastic programs with recourse, chance constrained programs, computational solution methods, approximation and sampling methods, and applications. Knowledge of linear programming, probability and statistics are required, while programming ability and knowledge of integer programming are helpful.

Prerequisite: MIE262, APS1005 or equivalent, and MIE231, APS106S or equivalent.

Overview

The aim of stochastic programming and robust optimization is to find optimal decisions in problems which involve uncertain data. These fields are currently developing rapidly with contributions from many disciplines including operations research, mathematics, probability, statistics and computer science. Conversely, they are being applied in a wide variety of subjects ranging from agriculture to financial planning and service systems server scheduling to hydrothermal generation planning. We will make a broad overview of the main themes and methods of the subject. Since stochastic programs are computationally very challenging, there will be a particular emphasis in this course on implementation and tools for solving stochastic programming instances.

Required background

Mathematical modeling background is required and will be assumed in this course. Key technical concepts will be briefly reviewed, but it is recommended that students have had a course in linear programming (or even better integer programming) before taking this course. A basic background in probability and statistics will be useful for parts of the course. Many assignments will require implementing algorithms, so programming ability (e.g., Python, Julia, C++ and Java) and familiarity with mixed-integer programming solvers (e.g., Cplex and Gurobi), or willingness and ability to learn, are also required.

Objectives

- Learn the terms, basic capabilities, and limitations of stochastic programming and robust optimization models.
- Learn to formulate stochastic programming and robust optimization models.
- Learn to implement the algorithms used to solve stochastic programming and robust optimization problems.
- Learn principles of decomposition algorithms for solving large-scale optimization problems.

Course web sites

The primary course website can be found in Quercus. This is where lecture slides and homework assignments will be posted. We will use Quercus also for posting your grades. Please check here that your grades correctly match the grade given on your assignments/exams. You may also be asked to submit assignments electronically through this web site.

The discussion website can be found in Piazza. Please sign up here (access code will be provided in the first lecture). When you have a question related to this course, you should first check Piazza to see if it has already been answered, and if not, post your question there. This way, all students benefit from your question and the answer. You are also allowed, and encouraged, to answer questions posted by other students.

Grading

The course grade will be based on the following components:

- 25% Homework
- 35% Final course project (due **December 9, 9am**, proposal is due November 2, 5pm)
- 20% Midterm exam, **November 1, 4pm**
- 20% Final exam, **December 13, 9am**

Students who have a conflict with the exam times should notify me by the second week of classes.

Homework

Homework assignments will be given roughly bi-weekly, and are a required part of the course. Most assignments will include a significant computational component. Students may work with a partner on homework assignments, but groups may not share answers with each other. (If we detect improper collaboration on an assignment, *all groups* involved will automatically receive a 0 on the assignment in question.) Please note that as the exams must be done independently, it is in each student's best interest to take an active role in all homework exercises.

Late assignments will be penalized 50% of the possible points for each day that it is late. So an assignment can receive no credit if it is two days late.

Classroom participation

Students are expected to attend lectures and are encouraged to participate by asking and answering questions. Students are also expected to follow standard classroom etiquette, including coming to class on time, and not causing classroom disruptions by talking during class, repeatedly checking your phone during class, sleeping, leaving early, etc.

Course project

Students are required to do a course project in groups of *at most* two people.

Project topics must be approved by me. I may be able to provide some suggestions on project topics, or you can choose a topic that is interesting to you, e.g., it may be related to your research. Students must submit to me a proposed project topic by **November 2, 5pm**, although it is strongly advised to start on the project earlier than this deadline.

Most projects will involve some sort of computational implementation. *The quality of the work should be close to that of a conference publication or (short) archival journal publication.* Besides the actual implementation, projects will consist of a written report that describes the model, algorithm, and computational results. An example of a strong project would be one that models a problem as a stochastic program and includes investigation of some advanced/specialized (even new!) methods for solving the problem. A few example projects might be:

- Development of a stochastic programming model that is relevant to your research, and implementation and comparison of method(s) to solve it. Publishing test instances in a "standard format" (e.g., as Python or Julia/JuMP models with associated data files) would be a great component of such a project.
- Implementation of an algorithm for multi-stage stochastic programming, e.g., a linear decision rule or approximate dual dynamic programming. (It can be for a specific structured problem – does not have to be a general purpose implementation).
- Implementation of a model for distributionally robust stochastic optimization, and comparison of solutions with a model that assumes the distribution is known. (Can be for a particular model.)
- Implementation of a branch-and-cut algorithm for solving stochastic integer programs, using integer programming based cuts to strengthen the scenario relaxation. (Can be for a particular model.)
- Development and comparison of stochastic programming models and robust optimization models for a particular problem.

Academic integrity

Of course I do not expect this to be a problem in a Ph.D. course, but academic misconduct will not be tolerated in this course. Please make yourself familiar with the University academic conduct guidelines: http://academicintegrity.utoronto.ca.

In this course, this mainly means that you should not copy anyone else's exam or allow anyone to copy your exam, and that assignment solutions should be your own work. Assignments may be done with a partner, **but groups must work independently.** In any assignment or exam, you must properly **give credit to any outside resources you use** (such as books, papers, web sites, other people, etc.). It is not allowed to use assignment solutions from previous semesters or other sources. Please do not hesitate to ask me if you have any uncertainty about what would be considered academic misconduct in this course.

Tentative list of topics

Following is a tentative list of topics which we will cover in this course.

- **Modeling.** Basic stochastic programming modeling concepts. Formulating and solving the extensive form of stochastic programs with recourse. Some statistics: value of stochastic solution and expected value of perfect information. Risk measures.
- **Algorithms.** Basic theory and properties of two-stage stochastic linear programs with recourse. Algorithms: Benders (L-Shaped) method (single and multi-cut), trust-region methods, level-method, progressive hedging.
- **Sampling and Monte Carlo Methods.** Review of statistics. Sample average approximation (SAA). Statistical inference on lower bounds, upper bounds and optimality gaps.
- **Robust Optimization.** Tractable (convex) optimization problems. Robust counterparts of uncertain LPs: basic theory, duality in robust optimization, choice of uncertainty sets.
- **Multistage Stochastic Programming.** Scenario tree based modeling. Two-stage approximations. Decision rule approximations. Stochastic dual dynamic programming.
- **Multistage Robust Optimization.** Adjustable robust counterparts of uncertain LPs.
- **Stochastic Integer Programming (time permitting).** Structure of value function. Algorithms for two-stage stochastic IP: dual decomposition and integer L-Shaped method. Valid inequalities for two-stage stochastic IP problems.
- **Uncertainty in Constraints (time permitting).** Basic theory of chance constraints and solvable cases. Analytic (and tractable) approximations of chance constraints. Sampling methods. Integer programming approaches to solving finite support problems.